**Python** **strings**

* **Python strings**
* **Slicing strings**
* **Modify strings**
* **Concatenate strings**
* **Format strings**
* **Escape characters**
* **String methods**
* **Python strings**
* Strings
* Assign a string to a variable
* Multiline strings
* Strings are arrays
* Looping through a string
* String length
* Check string
* Check if not
* **Slicing strings**
* Slicing
* Slice from the start
* Slice to the end
* Negative indexing
* **Modify strings**
* Upper case
* Lower case
* Remove white space
* Replace string
* Split string
* Sting methods
* **Concatenate strings**
* **Format strings**
* **Escape characters**
* **String methods**
* Strings in python are surrounded by either single quotation marks, or double quotation marks.
* 'Hello' is the same as "hello".
* You can display a string literal with the print() function:
* Assigning a string to a variable is done with the variable name followed by an equal sign and the string:
* You can assign a multiline string to a variable by using three quotes.
* You can use three double quotes Or three single quotes:
* Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
* However, Python does not have a character data type, a single character is simply a string with a length of 1.
* Square brackets can be used to access elements of the string.
* Get the character at position 1 (remember that the first character has the position 0):
* Since strings are arrays, we can loop through the characters in a string, with a for loop.
* To get the length of a string, use the len() function.
* The len() function returns the length of a string:
* To check if a certain phrase or character is present in a string, we can use the keyword in.
* To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.
* You can return a range of characters by using the slice syntax.
* Specify the start index and the end index, separated by a colon, to return a part of the string.
* By leaving out the start index, the range will start at the first character:
* By leaving out the end index, the range will go to the end:
* Use negative indexes to start the slice from the end of the string:
* Python has a set of built-in methods that you can use on strings.
* The upper() method returns the string in upper case.
* The lower() method returns the string in lower case.
* Whitespace is the space before and/or after the actual text, and very often you want to remove this space.
* The replace() method replaces a string with another string.
* The split() method returns a list where the text between the specified separator becomes the list items.
* The split() method splits the string into substrings if it finds instances of the separator.
* To concatenate, or combine, two strings you can use the + operator.
* we cannot combine strings and numbers .
* But we can combine strings and numbers by using the format() method!
* The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:
* Use the format() method to insert numbers into strings.
* The format() method takes unlimited number of arguments, and are placed into the respective placeholders:
* You can use index numbers {0} to be sure the arguments are placed in the correct placeholders.
* To insert characters that are illegal in a string, use an escape character.
* An escape character is a backslash \ followed by the character you want to insert.
* An example of an illegal character is a double quote inside a string that is surrounded by double quotes.
* You will get an error if you use double quotes inside a string that is surrounded by double quotes:
* To fix this problem, use the escape character \":
* The escape character allows you to use double quotes when you normally would not be allowed.

## **Escape Characters:**

Other escape characters used in Python:

|  |  |
| --- | --- |
| Code | Result |
| \' | Single Quote |
| \\ | Backslash |
| \n | New Line |
| \r | Carriage Return |
| \t | Tab |
| \b | Backspace |
| \f | Form Feed |
| \ooo | Octal value |
| \xhh | Hex value |

* Python has a set of built-in methods that you can use on strings.
* 1 String methods return new values. They do not change the original string.
* String methods:

|  |  |
| --- | --- |
| Method | Description |
| capitalize() | Converts the first character to upper case |
| casefold() | Converts string into lower case |
| center() | Returns a centered string |
| count() | Returns the number of times a specified value occurs in a string |
| encode() | Returns an encoded version of the string |
| endswith() | Returns true if the string ends with the specified value |
| expandtabs() | Sets the tab size of the string |
| find() | Searches the string for a specified value and returns the position of where it was found |
| format() | Formats specified values in a string |
| format\_map() | Formats specified values in a string |
| index() | Searches the string for a specified value and returns the position of where it was found |
| isalnum() | Returns True if all characters in the string are alphanumeric |
| isalpha() | Returns True if all characters in the string are in the alphabet |
| isdecimal() | Returns True if all characters in the string are decimals |
| isdigit() | Returns True if all characters in the string are digits |
| isidentifier() | Returns True if the string is an identifier |
| islower() | Returns True if all characters in the string are lower case |
| isnumeric() | Returns True if all characters in the string are numeric |
| isprintable() | Returns True if all characters in the string are printable |
| isspace() | Returns True if all characters in the string are whitespaces |
| istitle() | Returns True if the string follows the rules of a title |
| isupper() | Returns True if all characters in the string are upper case |
| join() | Joins the elements of an iterable to the end of the string |
| ljust() | Returns a left justified version of the string |
| lower() | Converts a string into lower case |
| lstrip() | Returns a left trim version of the string |
| maketrans() | Returns a translation table to be used in translations |
| partition() | Returns a tuple where the string is parted into three parts |
| replace() | Returns a string where a specified value is replaced with a specified value |
| rfind() | Searches the string for a specified value and returns the last position of where it was found |
| rindex() | Searches the string for a specified value and returns the last position of where it was found |
| rjust() | Returns a right justified version of the string |
| rpartition() | Returns a tuple where the string is parted into three parts |
| rsplit() | Splits the string at the specified separator, and returns a list |
| rstrip() | Returns a right trim version of the string |
| split() | Splits the string at the specified separator, and returns a list |
| splitlines() | Splits the string at line breaks and returns a list |
| startswith() | Returns true if the string starts with the specified value |
| strip() | Returns a trimmed version of the string |
| swapcase() | Swaps cases, lower case becomes upper case and vice versa |
| title() | Converts the first character of each word to upper case |
| translate() | Returns a translated string |
| upper() | Converts a string into upper case |
| zfill() | Fills the string with a specified number of 0 values at the beginning |